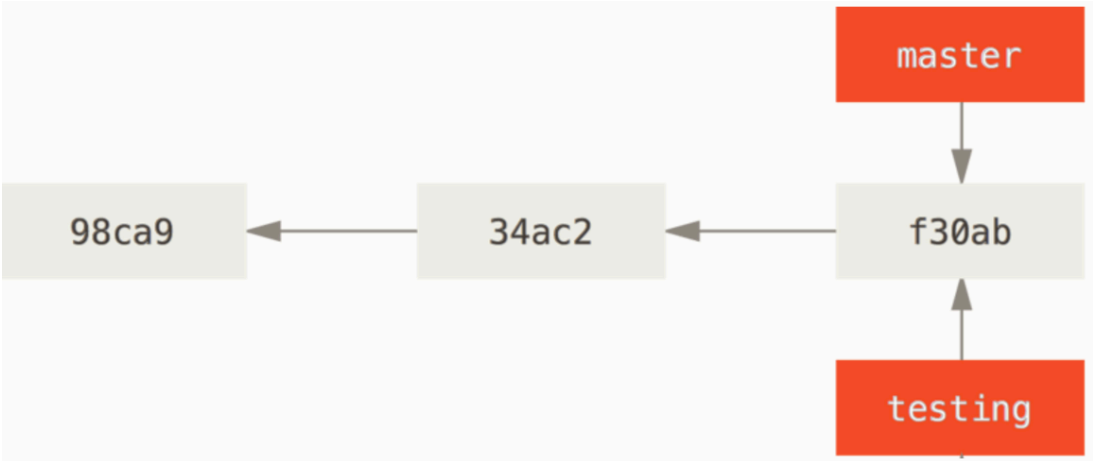
# Open Source Software

Kookmin University
Department of Computer Science

# Git Branch

- Branch (가지)
  - Diverging from master and continues to work without interrupting the master
- In most cases, master branch is the default branch that keeps track of the official release
- If you want to work on some features, you have to make a branch for the purpose

# Git Branch




- To create a branch
  - git branch testing
- To check which branch you are in
  - git branch

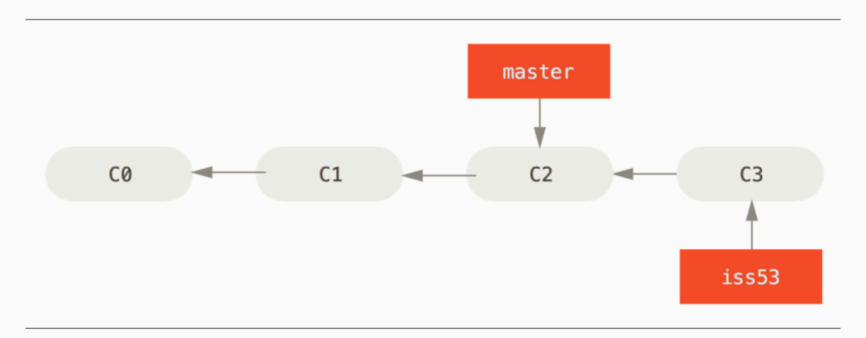
```
Kyungyongs-MacBook-Pro:oss kyungyonglee$ git branch testing
Kyungyongs-MacBook-Pro:oss kyungyonglee$ git branch
* master
  readme-setup
  testing
Kyungyongs-MacBook-Pro:oss kyungyonglee$
```

- To change a branch
  - git checkout branch-name

```
Kyungyongs-MacBook-Pro:oss kyungyonglee$ git checkout testing
Switched to branch 'testing'
Kyungyongs-MacBook-Pro:oss kyungyonglee$ git branch
  master
  readme-setup
* testing
Kyungyongs-MacBook-Pro:oss kyungyonglee$
```

# Merging Between Branches

- git merge DEST_BRANCH
  - Gets commits from the DEST_BRACNCH to the current branch



# Exercise: Merging Between Branches

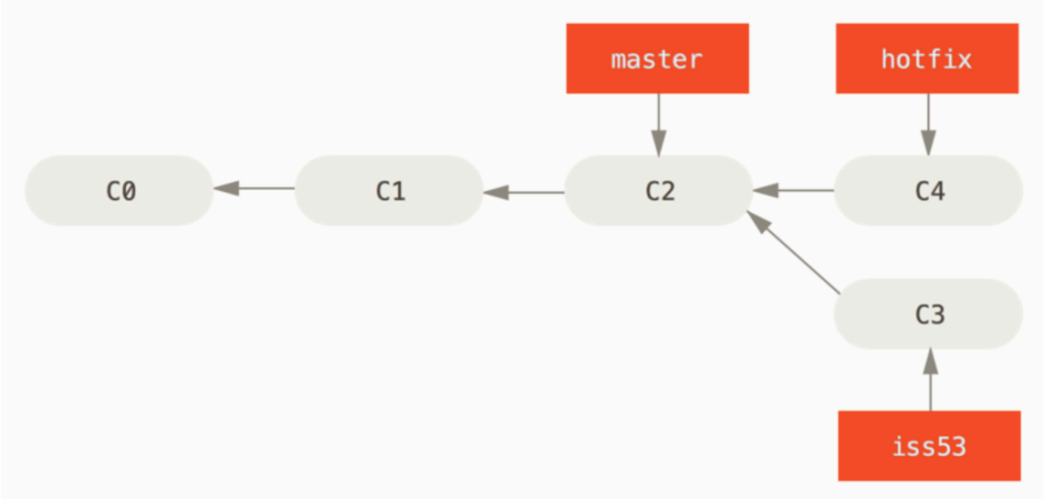
- Work in your oss-2016-homework repository
- Make sure you are in the master branch
  - git checkout master
- Create a file with a name of “merge-file” and write your name in the first line and your slack id in the second line
  - vi merge-file
- Commit the change in the master branch
- Create a branch “merge-test-1”
  - git checkout -b merge-test-1
  - ls should show merge-file that was created in the master branch
- Open merge-test file using and add a message “Open Source Software” using vi editor

# Exercise Continued

- In the merge-test-1 branch, commit the change that was made to merge-file
  - git add merge-file; git add → You will see a VI editor to type commit message
- Check the log
  - git log -3 // show only last three commits
- Check out to master branch and current commits
  - git checkout master; git log -3
- Now merge changes from merge-test-1 to master branch
  - git merge merge-test-1
  - git log -3

# Conflicts Between Branches

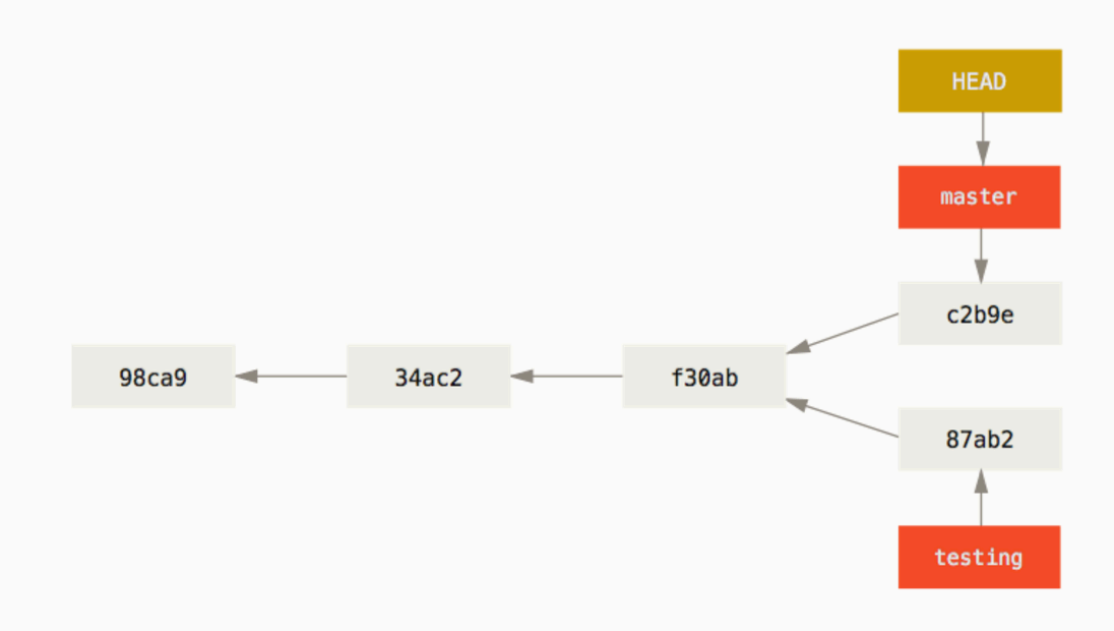
- Changes from different branches can result in conflict




- In the picture, everyone works on the snapshot of C2, and C3 was pushed. Then C4 tries to push
- The C4 change is based on C2. However, the current master is based on C3.
- The work of C4 should be done again based on C3: Update!

# Branch Diverging

- Branches can diverge as different works are done on the branches separately
- You can move between branches using
  - git checkout
  - git branch
- Branch is a very lightweight
  - Strength of Git

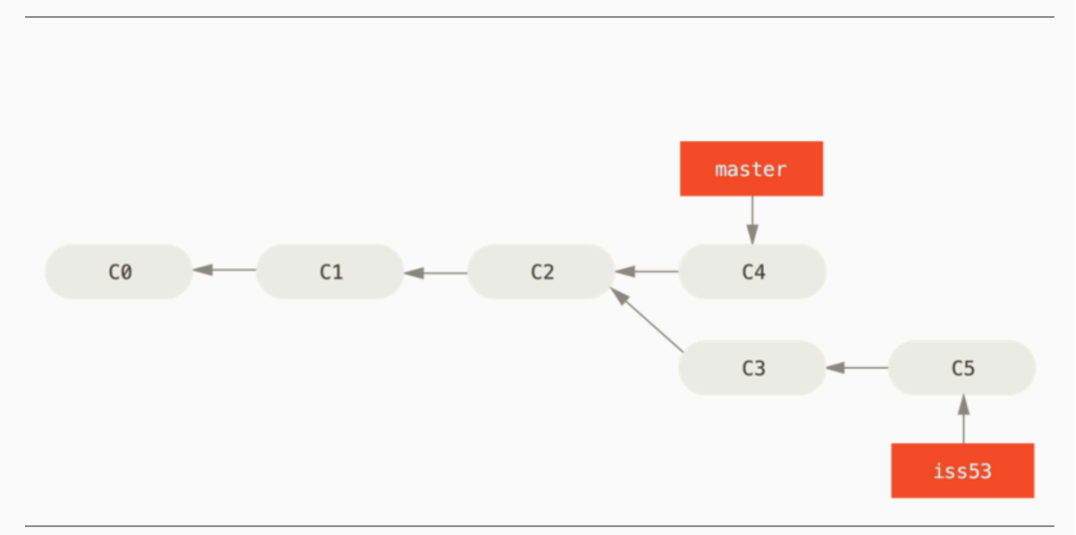


# Branch Work Scenario

- Developer A works on a cool feature on a branch name cool
- The developer gets a phone call reporting a critical issue in the current product
- The developer stops working on the cool feature and make commits
- The developer changes a branch to fix the issue
- After fixing the issue, merge to the master branch
- Later, the developer comes back to the cool branch and continues the cool work

# Merging Both Modified Branches

- In merging two branches
  - Easy if only one branch is updated
- Both branches are modified
  - No conflicts
  - Conflicts



# Let's Diverge Branch without Conflict

- From the master branch, create merge-test-2 branch
  - git checkout –b merge-test-2
- Open merge-file using vi editor, add a line of "change from merge-test-2" at the end of merge-file
- Commit the change to merget-test-2 branch
- Checkout to merge-test-1 branch
  - git checkout merge-test-1
- Open merge-file using vi editor, add a line of "change from merge-test-1" at the beginning of merge-file
- Commit the change to merget-test-1 branch

# Merging in Master Branch

- Checkout to master branch
  - git checkout master
- Merge from merge-test-2
  - git merge merge-test-2
- Check if the merge is successful
  - git log -3; vi merge-file
- Merge from merge-test-1
  - git merge merge-test-1
  - It will direct you to vi editor forcing you merge → here is where the version conflict happens
  - Store the file (using :wq)
- Check if the merge is successful
  - git log -3; vi merge-file
  - git log --graph

# Git Merge with Conflict

- Checkout to merge-test-1
- As the master branch is already updated, reflect the change
  - git merge master
- Add “merge successful from merge-test-1” at the end of merge-test
- Commit the change
  - Git add …; git commit …;
- Checkout to merge-test-2
- As the master branch is already updated, reflect the change
  - git merge master
- Add “merge successful from merge-test-2” at the end of merge-test
- Commit the change

# Now Let's Merge From Master Branch

- Both merge-test-1 and merge-test-2 modified the end of merge-test file!!
- Move to the master branch
  - git checkout master
- Merge merge-test-1
  - git merge merge-test-1
  - Check if the merge is successful
    - git log;cat merge-file
- Merge merge-test-2
  - git merge merge-test-2

# Now There is a Conflict

- Both merge-test-1 and merge-test-2 modified the last line of merge-file

```
Kyungyongs-MacBook-Pro:oss-2016-homework kyungyonglee$ git merge merge-test-2
Auto-merging merge-file
CONFLICT (content): Merge conflict in merge-file
Automatic merge failed; fix conflicts and then commit the result.
Kyungyongs-MacBook-Pro:oss-2016-homework kyungyonglee$
```

- Check which file has conflicts
  - git status

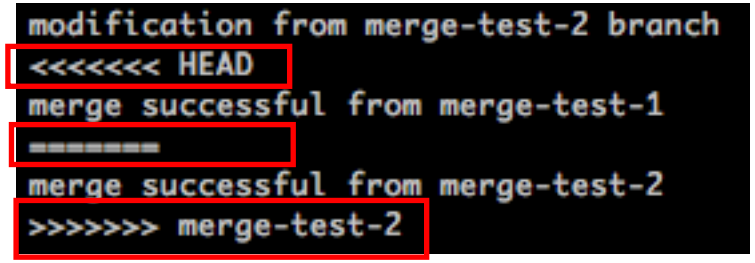
```
Kyungyongs-MacBook-Pro:oss-2016-homework kyungyonglee$ git status
On branch master
Your branch is ahead of 'origin/master' by 7 commits.
  (use "git push" to publish your local commits)
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   merge-file
```

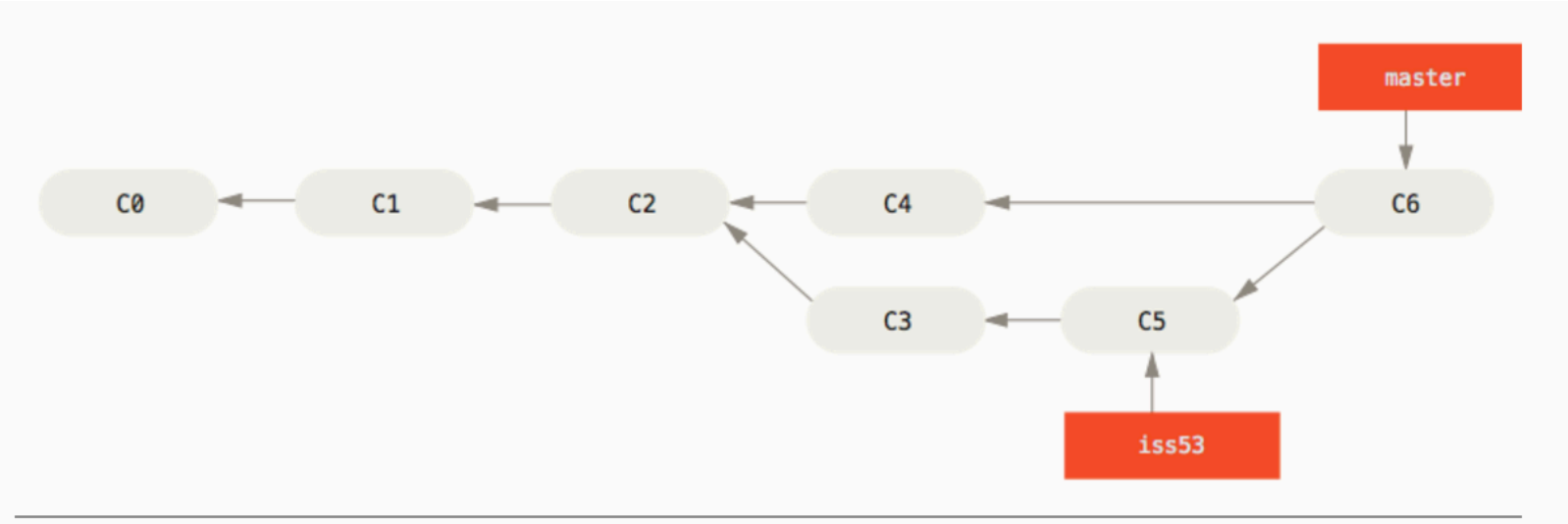
# Now Resolve the Conflicts

- Open merge-file
  - vi merge-file
  - <<<<<<< HEAD
    - The current value
  - =======
    - End of current value
  - >>>>>>> merge-test-2
    - Change from merge-test-2 branch
  - Manually choose one that you think is valid, and store
  - Add the change and commit
    - git add merge-file;git commit -m “done for conflict resolve”



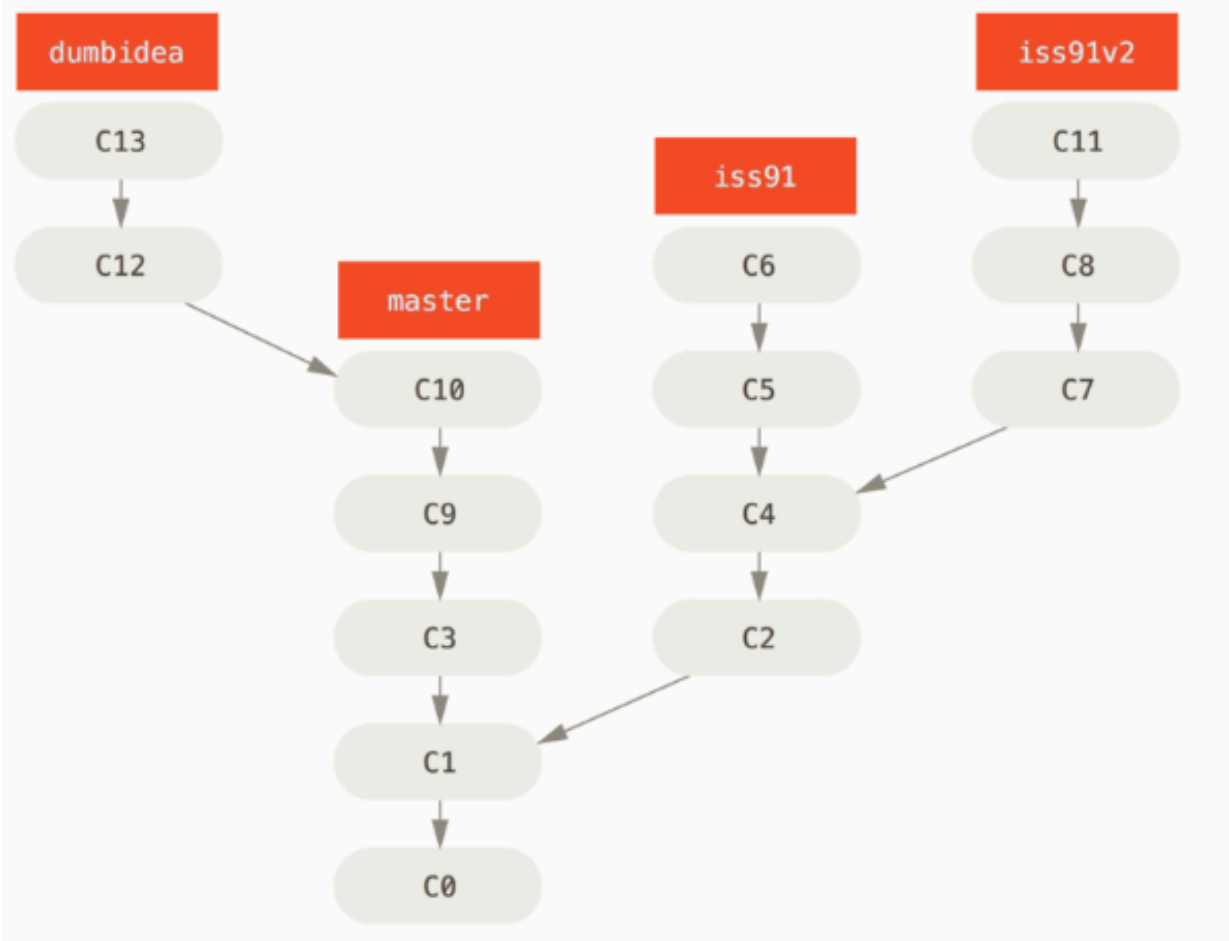
# Merging Two Branches

- Creates a new commit after merging changes from both branches



# Using Branch Best Practices

- Merging between branch is easy
  - Might not be true for you now but at some point you will….
- In general, there is one main branch - master.
- Multiple work branches for different issues and topics



# Only Local Changes So Far

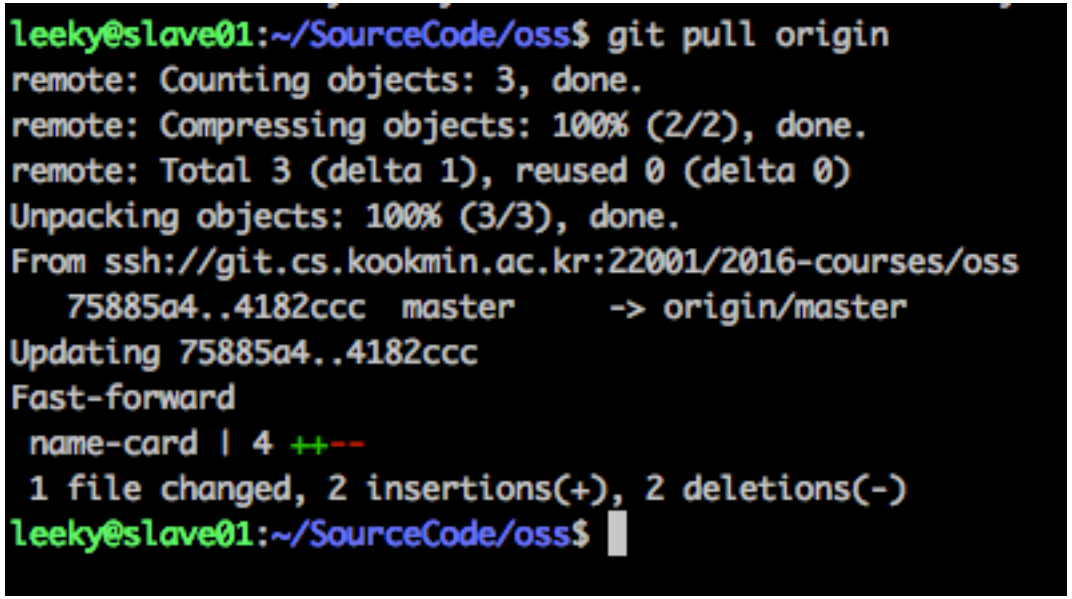
- Thus far, we worked only on the local branches.
- Note that by pushing to origin, you can upload your change (commits) to a remote server

# Update a Branch from Remote - Fetch

- git fetch remote_name
  - It goes to the remote branch and pulls down all the data from the remote project
  - git fetch origin – download the current version from the origin
  - It only downloads the change locally – not really updating
- git pull remote_name
  - It fetches the newest changes from the remote origin and update your local copies (fetch + merge)

# Git Fetch/Pull

- Now move to oss repository
- Update from a master branch
  - git pull origin



# Let's Modify the File All Together

- In the master branch
- In the name-card file, add your class – slack ID - name
  - EX: C1 – leeky – Kyungyong
- Add the file and commit
  - git add name-card
  - git commit –m "adding my name"
- Now try to push to remote
  - git push origin master
  - See what happens
    - You have to fetch the change from other students
    - Resolve conflict → DO NOT SIMPLY REMOVE OTHERS' NAME
    - Add/Commit/Push until it succeeds…

# Reference

- eecs.mines.edu/Courses/csci306/CHAPTERS/Git/GitIntro.ppt
- ProGit - http://www.git-scm.com/book/en/v2